Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71220876**
Nama Lengkap	**Devina Elisse Putri**
Minggu ke / Materi	**04 / Modular Progamming**

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

MATERI 1

Dalam program, fungsi-fungsi seperti input() dan print() adalah bagian dari Python yang sudah tersedia (built-in function), digunakan untuk interaksi dengan pengguna dan menampilkan output. Fungsi-fungsi ini merupakan bagian dari konsep modular programming, di mana program dibagi menjadi bagian-bagian (modul) yang memiliki tujuan khusus dan dapat digunakan ulang. Ada dua jenis fungsi: bawaan (built-in function) dan yang dibuat sendiri oleh programmer. Contohnya adalah fungsi tambah() untuk menghitung jumlah dua bilangan.

Fungsi tambah() ditandai dengan:

- Keyword 'def' digunakan untuk mendefinisikan fungsi.

- Nama fungsi: tambah().

- Isi fungsi diperlukan untuk di-indentasi.

- Fungsi memerlukan dua argumen (parameter a dan b).

- Hasil penjumlahan disimpan dalam variabel, menggunakan 'return' untuk mengembalikan nilai.

```
# definisikan fungsi tambah terlebih dahulu
def tambah(a, b):
    hasil = a + b
    return hasil

# panggil fungsi tambah dengan dua arguments berupa nilai: 10 dan 5
c = tambah(10, 5)
print(c)
```

```
15
```

Jalannya program tersebut:

- Baris 1 adalah komentar.

- Baris 2-4 mendefinisikan fungsi tambah(), belum dijalankan sampai dipanggil.

- Baris 5-6 adalah baris kosong/komentar diabaikan.

- Baris 7: variabel c diisi hasil dari pemanggilan fungsi tambah() dengan argumen 10 dan 5.

- Program melompat ke baris 2 saat fungsi dipanggil, dengan a=10 dan b=5.

- Lanjut ke baris 3, variabel hasil berisi 10 + 5 = 15.

- Di baris 4, 'return' mengembalikan nilai hasil (15), fungsi selesai.

- Kembali ke baris 7, c diisi 15 (hasil fungsi).

- Baris 8 menampilkan nilai c (output: 15).

Jadi, urutan jalannya program adalah: 1-2-3-4-5-6-7-2-3-4-7-8.

MATERI 2

```
def tambah(a, b, c):
   hasil = a + b + c
   return hasil

nilai1 = 70
nilai2 = 85
nilai3 = 55

rata_rata = tambah(nilai1, nilai2, nilai3)/3
print(rata_rata)
```

```
70.0
```

Program tersebut menjalankan fungsi `tambah()` dengan langkah-langkah sebagai berikut:
1. Definisi fungsi `tambah(a, b, c)`, namun belum ada yang dijalankan.
2. Variabel `nilai1`, `nilai2`, dan `nilai3` didefinisikan dan diisi dengan nilai masing-masing.
3. Fungsi `tambah()` dipanggil dengan tiga argumen, yaitu `nilai1`, `nilai2`, dan `nilai3`.
4. Program berpindah ke dalam fungsi `tambah()`, di mana nilai-nilai parameter diisi dengan argumen yang diterima.
5. Variabel `hasil` diisi dengan hasil penjumlahan dari ketiga parameter.
6. Nilai `hasil` dikembalikan menggunakan pernyataan `return`.
7. Program kembali ke baris di mana fungsi `tambah()` dipanggil dan menyimpan hasilnya dalam variabel `rata_rata`.
8. Nilai dari `rata_rata` kemudian ditampilkan.
Jadi, program tersebut menghitung rata-rata dari tiga nilai yang diberikan dengan menggunakan fungsi `tambah()` dan menampilkan hasilnya.

MATERI 3

```
def hitung_belanja(belanja, diskon=0):
    bayar = belanja - (belanja * diskon)/100
    return bayar
print(hitung_belanja(100000))
print(hitung_belanja(100000, 10))
print(hitung_belanja(100000, 50))
```

```
100000.0
90000.0
50000.0
```

Fungsi dapat memiliki parameter opsional dengan nilai default yang sudah didefinisikan sebelumnya. Contohnya, fungsi `hitung_belanja()` memiliki parameter `belanja` dan `diskon`, dengan `diskon` memiliki nilai default 0 (0%).

```
def cetak(a, b, c):
  print("Nilai a: ",a)
  print("Nilai b: ",b)
  print("Nilai c: ",c)
cetak(20, 30, 40)
```

```
Nilai a:  20
Nilai b:  30
Nilai c:  40
```

MATERI 4

Anonymous function dalam Python adalah fungsi tanpa nama, yang merupakan fitur tambahan dan bukan fitur utama. Ini berbeda dengan bahasa pemrograman fungsional seperti Haskell, Lisp, dan Erlang.

```
def tambah(a, b):
  hasil = a + b
  return hasil
print(tambah(10,20))
```

```
30
```

KEGIATAN PRAKTIKUM 1

Buatlah sebuah fungsi yang dapat menghitung tagihan listrik seseorang (pasca bayar) dengan beberapa informasi berikut ini:

• Jumlah pemakaian (dalam kwh)

• Golongan tarif (1 - 4). Diasumsikan input golongan tarif selalu valid.

• – Golongan 1: Rp. 1500/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 2000/kwh. – Golongan 2: Rp. 2500/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 3000/kwh. – Golongan 3: Rp. 4000/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 5000/kwh. – Golongan 4: Rp. 5000/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 7000/kwh. Jika tidak ada informasi golongan tarif, maka diasumsikan menggunakan tarif golongan 3. Definisikan fungsi tersebut!

■ Ada beberapa hal yang perlu diperhatikan sebelum anda mendefinisikan fungsi untuk menghitung tagihan listrik tersebut:

• Fungsi membutuhkan dua parameter, yaitu jumlah pemakaian dan golongan tarif.

• Jumlah pemakaian menggunakan tarif yang berbeda untuk 100 kwh pertama, dan setelah 100 kwh.

• Golongan tarif memiliki nilai default = 3, sehingga harus didefinisikan sebagai optional argument.
Fungsi untuk menghitung tagihan listrik dan contoh penggunaannya dapat dilihat pada kode program berikut ini:

```
def tagihan_listrik(pemakaian, golongan=3):
 bayar = 0
 pemakaian_100 = 100 if pemakaian > 100 else pemakaian
 pemakaian_100_lebih = pemakaian - pemakaian_100
 if golongan == 1:
  bayar = pemakaian_100 * 1500 + pemakaian_100_lebih * 2000
 elif golongan == 2:
  bayar = pemakaian_100 * 2500 + pemakaian_100_lebih * 3000
 elif golongan == 3:
  bayar = pemakaian_100 * 4000 + pemakaian_100_lebih * 5000
 elif golongan == 4:
  bayar = pemakaian_100 * 5000 + pemakaian_100_lebih * 7000
 return bayar

 print(tagihan_listrik(130))
 print(tagihan_listrik(80, 4))
 print(tagihan_listrik(golongan=1, pemakaian=175))
```

PRAKTIKUM 2

Python menggunakan prinsip yang disebut Call-by-Object untuk pengiriman parameter. Perlakuan terhadap parameter bergantung pada apakah argumen yang diberikan bersifat immutable atau tidak. Jika argumen bersifat immutable, seperti integer, string, dan tuple, nilai argumen tidak dapat diubah oleh fungsi. Sebagai contoh, pada fungsi `abc()` yang menerima tiga argumen integer, nilai-nilai tersebut

```
def abc(a, b, c):
  a = b + c
  b = c + a
  c = a + b
nilai1 = 20
nilai2 = 30
nilai3 = 40
abc(nilai1, nilai2, nilai3)
print(nilai1)
print(nilai2)
print(nilai3)
```

```
20
30
40
```

bersifat immutable, sehingga tidak akan terpengaruh oleh perubahan nilai parameter dalam fungsi `abc()`.

PRAKITIKUM 3

```
def kelipatan_sembilan(angka):
    if angka % 9 == 0:
        return True
    else
        return False
```

Ubahlah fungsi tersebut menjadi bentuk lambda function!

Setelah diubah akan menjadi :

```
kelipatan_sembilan = lambda angka: angka % 9 == 0
```

Lambda function dalam Python terdiri dari tiga bagian:
- Keyword: `lambda`
- Bound variable: `angka`
- Body: Pengecekan apakah `angka` habis dibagi 9 atau tidak (kelipatan 9).

Dan bila diaplikasikan akan menjadi :

```
kelipatan_sembilan = lambda angka: angka % 9 == 0
print(kelipatan_sembilan(81))
print(kelipatan_sembilan(2000))
```

```
True
False
```

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

SOAL 1

```
[5]  #no 1 laprak
     def cek_angka(a, b, c):
         # Cek apakah nilai a, b, c berbeda semua
         if a != b and a != c and b != c:
             # Cek kemungkinan jika jumlah dua parameter sama dengan parameter yang tersisa
             if a + b == c or a + c == b or b + c == a:
                 return True
             else:
                 return False
         else:
             return False
```

1. Fungsi cek_angka(a, b, c) adalah fungsi yang mengambil tiga parameter: a, b, dan c.

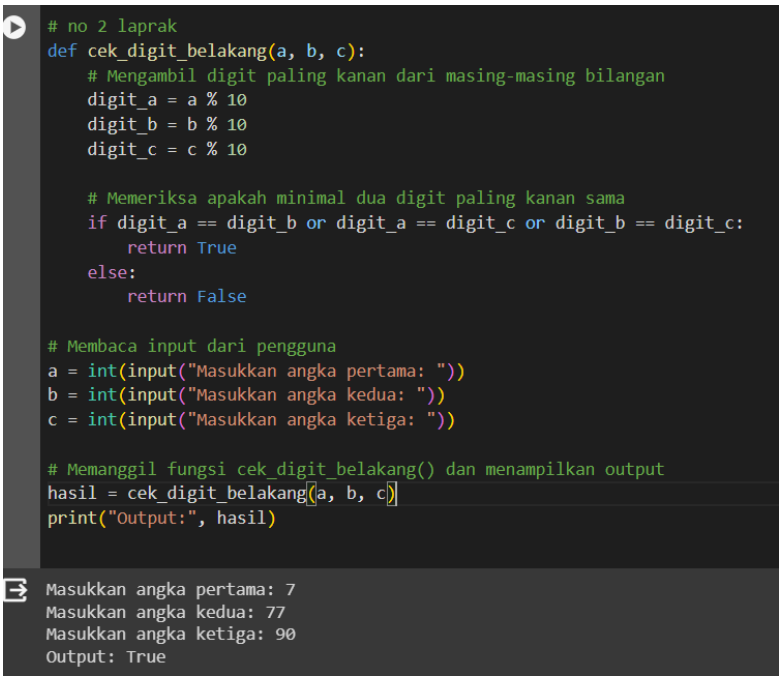
2. Dalam fungsi ini, kita ingin memeriksa apakah ketiga parameter memiliki nilai yang berbeda.

3. Jika a, b, dan c semua berbeda, maka kita periksa apakah ada dua parameter yang ketika dijumlahkan menghasilkan nilai yang sama dengan parameter ketiga.

4. Jika salah satu kondisi di atas terpenuhi, fungsi akan mengembalikan True, jika tidak, maka akan mengembalikan False.

Jadi, fungsi cek_angka() digunakan untuk memeriksa apakah ketiga angka yang diberikan memenuhi dua syarat: nilai yang berbeda semua, dan adanya kemungkinan bahwa dua angka yang dijumlahkan akan sama dengan angka yang tersisa.

SOAL 2



```
# no 2 laprak
def cek_digit_belakang(a, b, c):
    # Mengambil digit paling kanan dari masing-masing bilangan
    digit_a = a % 10
    digit_b = b % 10
    digit_c = c % 10

    # Memeriksa apakah minimal dua digit paling kanan sama
    if digit_a == digit_b or digit_a == digit_c or digit_b == digit_c:
        return True
    else:
        return False

# Membaca input dari pengguna
a = int(input("Masukkan angka pertama: "))
b = int(input("Masukkan angka kedua: "))
c = int(input("Masukkan angka ketiga: "))

# Memanggil fungsi cek_digit_belakang() dan menampilkan output
hasil = cek_digit_belakang(a, b, c)
print("Output:", hasil)
```

```
Masukkan angka pertama: 7
Masukkan angka kedua: 77
Masukkan angka ketiga: 90
Output: True
```

1. digit_a = a % 10, digit_b = b % 10, digit_c = c % 10 : Ini adalah langkah pertama dalam fungsi, di mana kita mengambil digit paling kanan dari masing-masing bilangan dengan menggunakan operator modulo % 10.

2. if digit_a == digit_b or digit_a == digit_c or digit_b == digit_c: : Ini adalah kondisi yang memeriksa apakah minimal dua digit paling kanan dari tiga bilangan sama.

3. Jika salah satu dari kondisi di atas terpenuhi, maka fungsi akan mengembalikan True, yang berarti minimal dua dari tiga bilangan memiliki digit paling kanan yang sama.

4. Jika tidak ada dua digit yang sama, maka fungsi akan mengembalikan False.

Fungsi ini digunakan untuk memeriksa apakah minimal dua dari tiga angka yang diberikan memiliki digit paling kanan yang sama. Setelah itu, program membaca input dari pengguna untuk tiga angka, memanggil fungsi cek_digit_belakang() dengan tiga angka tersebut, dan menampilkan output yang sesuai.

SOAL 3

```python
#no 3 laprak
# Celcius to Fahrenheit: F = (9/5) * C + 32
celcius_to_fahrenheit = lambda c: (9/5) * c + 32

# Celcius to Reamur: R = 0.8 * C
celcius_to_reamur = lambda c: 0.8 * c

# Test-case
c1 = 100
f1 = celcius_to_fahrenheit(c1)
print(f"Input C = {c1}. Output F = {f1}")

c2 = 80
r2 = celcius_to_reamur(c2)
print(f"Input C = {c2}. Output R = {r2}")

c3 = 0
f3 = celcius_to_fahrenheit(c3)
print(f"Input C = {c3}. Output F = {f3}")
```

```
Input C = 100. Output F = 212.0
Input C = 80. Output R = 64.0
Input C = 0. Output F = 32.0
```

1. `celcius_to_fahrenheit = lambda c: (9/5) * c + 32` : Ini adalah sebuah lambda function yang mengonversi suhu dari Celcius ke Fahrenheit. Lambda function ini mengambil parameter `c` (suhu dalam Celcius) dan mengembalikan hasil perhitungan `(9/5) * c + 32`, yang merupakan rumus konversi suhu dari Celcius ke Fahrenheit.

2. `celcius_to_reamur = lambda c: 0.8 * c` : Ini adalah lambda function lainnya yang mengonversi suhu dari Celcius ke Reamur. Lambda function ini juga mengambil parameter `c` dan mengembalikan hasil perhitungan `0.8 * c`, yang merupakan rumus konversi suhu dari Celcius ke Reamur.

3. Kemudian terdapat test-case untuk menguji kedua fungsi konversi yang telah dibuat:

- `c1 = 100`, maka `f1 = celcius_to_fahrenheit(c1)` akan menghasilkan konversi suhu dari 100 Celcius ke Fahrenheit.

- `c2 = 80`, maka `r2 = celcius_to_reamur(c2)` akan menghasilkan konversi suhu dari 80 Celcius ke Reamur.

- `c3 = 0`, maka `f3 = celcius_to_fahrenheit(c3)` akan menghasilkan konversi suhu dari 0 Celcius ke Fahrenheit

4. Setelah itu, hasil konversi suhu untuk masing-masing test-case ditampilkan menggunakan `print()` dengan format yang sesuai.

Jadi, pada kode di atas, lambda function digunakan untuk membuat fungsi-fungsi konversi suhu yang sederhana, kemudian digunakan untuk mengonversi suhu pada beberapa test-case yang telah diberikan, dan hasilnya ditampilkan.